Support for Aspectual Modeling to Multiagent System Architecture

Carla Silva[1], Márcia Lucena[1,2], Jaelson Castro[1], João Araújo[3], Ana Moreira[3], Fernanda Alencar[4]
[1]*Centro de Informática, Univ. Federal de Pernambuco, Recife, Brasil, 50740-540*
{ctlls,mjnrl,jbc}@cin.ufpe.br
[2]*Departamento de Informática, Universidade Federal do Rio Grande do Norte, Campus Universitário, Natal, Brasil*
marciaj@dimap.ufrn.br
[3]*Dept. de Informática, FCT, Universidade Nova de Lisboa, Caparica, Portugal, 2829-516*
{ja, amm}@di.fct.unl.pt
[4]*Dept. de Eletrônica e Sistemas, Univ. Federal de Pernambuco, Recife, Brazil, 50740-530*
fmra@ufpe.br

Abstract

A Multiagent system (MAS) architecture is structured in terms of autonomous and communicating components. Agent orientation does not support the modularization of some system properties that affect several system components. These properties are called "crosscutting concerns" and need to be explicitly captured in the architectural design of MAS. Aspect-orientation provides abstractions to identity and modularize crosscutting concerns throughout the software lifecycle. This work uses aspect-oriented abstractions to define a modeling language to describe and modularize crosscutting concerns in MAS architecture. An e-commerce example is used to illustrate the application of the proposed modeling language.

1. Introduction

Software architectures play a fundamental role in dealing with difficulties inherent to complex software systems development [6]. These software systems execute in a context where organizational processes are well established and, therefore, need to be built with flexible architectures, based on social and intentional concepts, to evolve consistently with its operational environment. In this context, Tropos [5] has defined a methodology to develop flexible systems based on agent technology.

Multiagent systems (MAS) are not free of crosscutting concerns, i.e., concerns that cut across the boundaries of other concerns. The separation of concerns principle states that each system concern must be handled separately from the others [8]. However, agent-oriented abstractions are not sufficient to modularize crosscutting concerns. In fact, several studies have identified crosscutting concerns at design level of MAS, such as control and distribution [4], design patterns [20], non-functional requirements (NFR) [3] and agent internal concerns [10]. Even in earlier phases of software development lifecycle, it is difficult to modularize certain system requirements by using traditional building blocks for requirements specification [1, 18]. As a result, some investigation has been performed to use aspect-orientation [11] in several abstraction levels of software specification to address this problem [1, 3, 4, 10, 13, 16, 17, 20, 21].

Aspect-oriented software development (AOSD) aims at handling crosscutting concerns explicitly, providing means for their systematic identification, modularization, representation and composition during all phases of the software development lifecycle [18]. Using AOSD to develop MAS encourages modular descriptions of such complex software by providing support for cleanly separating the system functionality from its crosscutting concerns. Our previous works proposed: (i) an aspect-oriented approach for MAS requirements specification [1]; (ii) an aspect-oriented modeling language for MAS detailed design [21], and; (iii) an aspect-oriented implementation of MAS [22] in the Tropos context.

Although several Architectural Description Languages (ADL) for aspect-oriented systems have been proposed in the literature [3, 13, 16, 17], none of them support the development of MAS using agent abstractions as first-class citizens. Therefore, there is a gap between aspect-oriented requirements and detailed design approaches for MAS. To close this gap and to be aligned with our previous work [21], we define our aspectual ADL for MAS as a UML (Unified Modeling Language) profile [14]. This language will allow (i) building MAS using aspectual components to

modularize crosscutting concerns, improving modularity, and (ii) using Model-Driven Development (MDD) [15] to perform model transformations from requirements specification to implementation code, improving traceability. The proposed language is illustrated by a classical case study for MAS: the Medi@ System [5].

This paper is organized as follows. Section 2 reviews MAS modeling. Section 3 introduces the aspectual profile as well as the aspect-oriented modeling language to describe MAS architectural design. Section 4 presents the Medi@ case study. Section 5 discusses related work. Finally, Section 6 summarizes our work and discusses future work.

2. Architectural Profile for MAS

Tropos [4] is a requirements-driven framework aimed at building MAS. According to it, MAS are composed of a number of SocialComponents (henceforth referred to as Component for simplicity) which depend on each other to achieve some Dependum. A Dependum defines an "agreement" of service provision between two Components. To support the architectural design in Tropos and benefit from UML tool support, a language to describe MAS architecture, called agency profile (Figure 1(a)), was defined in [18] as a UML profile.

Figure 1(a) shows only the elements from agency profile directly related to the aspectual profile (Section 3). Thus, the Component responsible for providing the service is related to the Dependum through a Dependee relationship. It possesses a SocialPort typed by the Dependum and playing the role of dependee. The Component which requests the service is related to the Dependum through a Depender relationship. It possesses a SocialPort typed by the Dependum and playing the role of depender. A Dependum can be of four types: goals, softgoals, tasks and resources [22], defined as the IntentionalKind enumeration. A Dependum is composed at least one SocialOperation which is implemented by the Component related to the Dependum through a Dependee relationship. This SocialOperation is directly involved in the achievement of the Dependum and to do this, components exchange messages through SocialPorts. A SocialConnector specifies a link that enables communication between two Components and must be attached, through SocialConnectorEnds, to exactly two SocialPorts.

The model in Figure 1(b) provides a candidate architecture for the situation where a Client requests some services (represented as the Dependums) to a Provider. The SocialOperation defined in the Dependum is performed by the Provider. Both the message for requesting the service execution and the message for confirming whether the service was successfully concluded are sent through the SocialConnector. The agency profile was inspired on the Tropos ontology [22] but it aims at being as generic as possible to be used by other MAS methodologies. Observe that although this profile has been defined to support MAS architecture description, agents should not appear as building blocks at this level of abstraction. Agents will only appear at the detailed design stage, when further detail related to agent abstractions and design patterns are to be added.

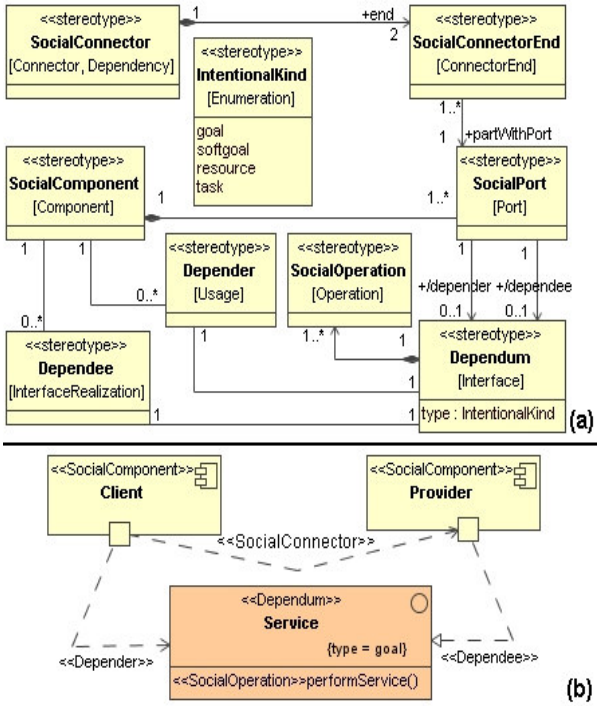

Figure 1. (a) Some elements of the Agency Profile; (b) An Architectural Model for MAS.

3. Aspectual Profile for MAS Architecture

To define an aspectual modeling language for MAS architecture, we extended the agency profile (Figure 1(a)) by defining new elements required to specify aspectual components, base components and the relationship between them.

When describing aspect-oriented abstractions we need to define (i) a unit to encapsulate the crosscutting concern – the Aspect component; (ii) each behavior a crosscutting concerns comprises – the Advice operation; (iii) the definition of the distinct ways a crosscutting concern may affect a base unit – the Enhancement interface; (iv) the point where the crosscutting concern should join the behavior of the base unit it cuts across – the BaseOperation operation;

(v) the specification of how and in which sequence the crosscutting behavior must be applied to the base units – the PointCut expression; (vi) the definition of a distinct interaction point between the crosscutting concern and the base unit – the AspectualPort port; and (vii) the explicit relationship between them – the Crosscut connector. The result is an Aspectual Profile (Figure 2) able to specify aspectual components and how they must be composed with the base unit(s) it cuts across. For simplicity, Figure 2 presents only two concepts (highlighted in gray color) defined in the agency profile (Figure 1(a)). By matching these concepts with the corresponding concepts in the agency profile, we can obtain the complete aspectual profile.

In Figure 2, an Aspect component encapsulates a crosscutting concern and each of its distinct behaviors is defined in an Enhancement interface provided by the Aspect. Each Aspect possesses at least one AspectualPort which specifies a distinct interaction point between that Aspect and the base components. Thus, an Aspect will affect a base component through a Crosscut connector which is attached to its AspectualPort by a CrosscutEnd and to the base component's port by a BaseConnectorEnd.

Observe that the base component affected by the Aspect can be either a Component (Figure 1(b)) or an Aspect, since the BaseConnectorEnd can be specialized in SocialConnectorEnd or CrosscutEnd. Considering the base component as a Component, the other extreme of the Crosscutt connector is attached to a SocialPort through a SocialConnectorEnd (see also Figure1 (a)). If the base component is an Aspect, the other extreme of the Crosscutt connector is also attached to an AspectualPort through a CrosscutEnd.

An important feature of the Crosscut connector is that it possesses the information related to the weaving between the Aspect and the base component. This information is a PointCut expression which maps an Advice operation (provided by the Aspect) to an Operation (performed by the base component) by using a CompositionRule (after, around or before), defined as an Enumeration metaclass. The PointCut expression informs the point and the situation in which the Aspect will affect the base component. Thus, the Crosscut connector contains expressions responsible for composing the Aspect to the Component. As specified in the UML metamodel [14], an expression can be placed in the form *symbol (operand1, operand2).*

Our profile defines the PointCut element as an expression in which the symbol is a CompositionRule, the operand1 is an Advice and the operand2 is a SocialOperation. It means that the operand1 will affect the operad2 in the situation stated by the symbol.

Both to describe the aspectual components and to enable the systematic composition of them with other components, we use the concept of model roles [12] to specialize specific classes of the aspectual Profile (Figure 2). Model roles are identified by preceding certain elements from the profile with a vertical bar, "|". Model roles have been used to describe and systematically apply object-oriented patterns [9], and agent-oriented patterns [20]. To help using our approach, let's consider the case study in Figure 3.

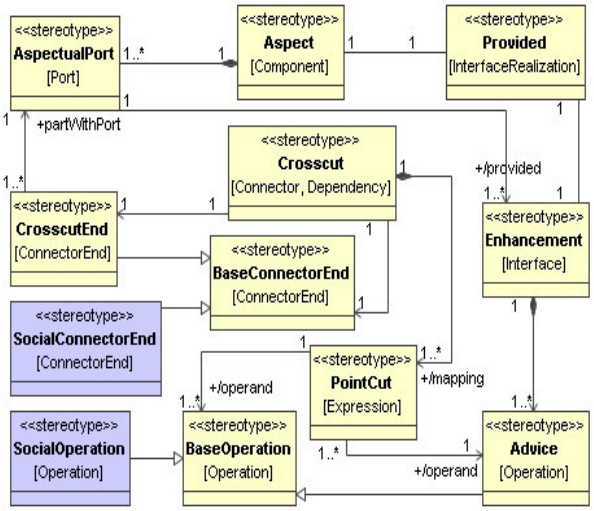

Figure 2. Aspectual Profile

4. Case Study

Medi@ system is an on-line store that allows customers to examine the media items in its catalogue and place orders [5]. Figure 3(a) shows the architectural design for Medi@ system, structured into three components: Front Store, that is responsible for supplying a customer with a web shopping cart to keep track of items the customer is buying when visiting *Medi@*; Back Store, that keeps track of all customers web information and other data of strategic importance to Media Shop, and; Billing Processor, that is in charge of the management of orders, bills and financial data.

To illustrate the separation of crosscutting concerns in the Medi@ architectural design, we have chosen one of the three NFRs elicited in Medi@ requirements [5] – the security concern. Clients exposed to the internet are, like servers, at risk in web applications. It is possible for web browsers and application servers to download or upload content and programs that could represent a certain degree of risk to the system and the information it manages. Thus, a security requirement might be operationalized by requesting client authorization, encrypting personal data or checking data consistency for each business transaction.

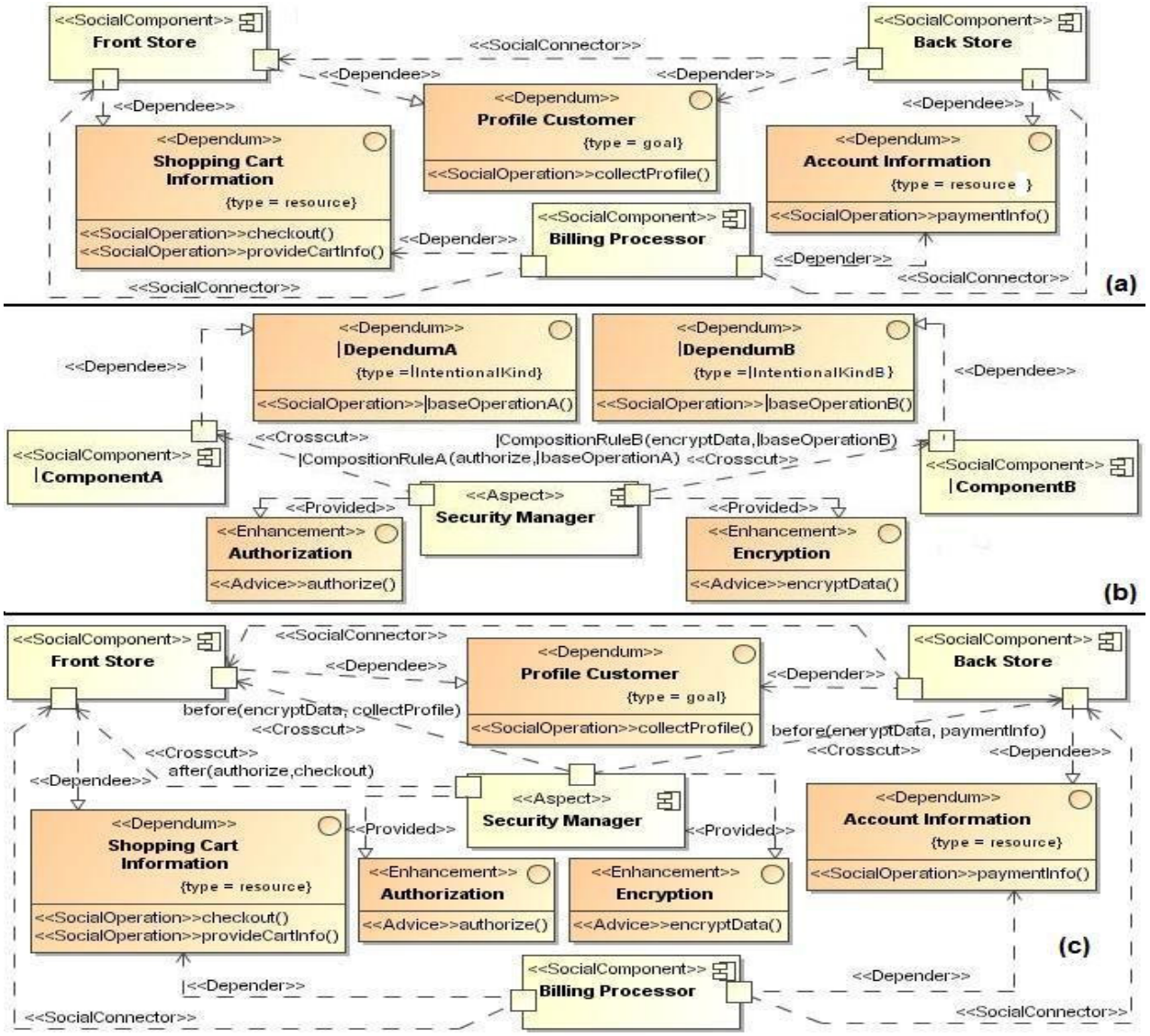

Figure 3. (a) Architecture for Medi@; (b) Security Aspect; (c) Aspectual Architecture for Medi@

The security concern is tangled and scattered with all Medi@ components. For example, when the customer places an order for the first time in the Medi@ system, he has to fulfill a profile form. It contains the Customer's personal data such as address, credit card number, phone number, etc. Thus, all the customer data might be encrypted by the Front Store Component before sending it to the Back Store Component. Besides, to close the order of items added to customer's shopping cart, the Front Store Component must request specific customer's information for authorization. Moreover, to process the billing information, it is needed to access customer's information such as the credit card number and delivery address. This data must be also encrypted by the Back Store Component before sending it to the Billing Processor Component.

To avoid that scattering and tangling of the security with base components of MAS, a proper modeling language must be used to locate this crosscutting concern in a separate component. Hence, we can use the aspectual modeling language to describe the Security aspect. In the model depicted in Figure 3(b), the Security aspect encapsulates two crosscutting behaviors: Encryption and Authorization. They are defined in each Enhancement interface provided by the Aspect, namely the Encryption and Authorization, which will affect differently each system component.

To make easier the understanding of the Security aspect description (Figure 3(b)), let's consider it affects

only SocialComponents. In this case, it is needed to specialize each metaclass of interest using model roles, i.e., the SocialComponent affected by the crosscut relationship, the Dependum realized by the affected component, the type of the dependum (of kind IntentionalElement) and the properties of the crosscut relationship: symbol attribute of type CompositionRule and the operand of type BaseOperation.

To compose the Security aspect (Figure 3(b)) with the Medi@ architectural design, we need to instantiate (or bind) the model roles present in the crosscut relationship and in the element affected by the crosscut relationship. For example, let us consider the Encryption enhancement. To compose it with the Medi@ architectural design (Figure 3(a)) the following bindings are made: Bind |ComponentB to Back Store; Bind |DependumB to Account Information; Bind |CompositionRuleB to before; Bind |baseOperationB to paymentInfo; Bind |IntentionalKindB to resource.

After composing the Security aspect to the Medi@ system, we can see the Encryption and Authorization enhancements affecting differently the components of Medi@ (See Figure 3(c)). For example, the Security aspect crosscuts the Front Store component with both the Encryption and Authorization enhancements, as well as the Back Store component with the Encryption enhancement. The Encryption enhancement adds to the Front Store component the ability to encrypt data before collecting the customer's profile. It also adds to the Back Store component the ability to encrypt data before sending payment information to the Billing Processor component. The Authorization enhancement, in turn, adds to the Front Store component the ability to authorize the customer after performing checkout of the shopping cart. This improved separation of concerns in MAS produces artifacts that are easier to maintain and reuse, since each concern in located into separate components. In fact, these benefits were captured through an empirical evaluation of the use of aspect orientation in the implementation of agent oriented design patterns reported in [22].

5. Related Work

Several languages have been proposed to describe architectural design of aspect-oriented systems. AspectualACME [3] proposal is based on enriching the composition semantics supported by architectural connectors instead of introducing new abstractions that add complexity into the architecture specification. AO-ADL [17] proposes to modularize both crosscutting and non-crosscutting behavior using the same component abstraction. A component is considered as being aspectual when it participates in an aspectual interaction. Thus, the distinction between aspectual and non-aspectual components is handled by the connector.

DAOP-ADL [16] defines aspects as first-order elements. Aspects' interfaces define both the messages that aspects are able to intercept and the events that aspects can capture in the base component. A set of rules define when and how the aspect behavior is executed. The main difference between our approach and the previous ones is that our modeling language supports aspect abstractions in MAS development.

AOGA [13] was developed to represent aspectual components as UML components with diamonds, which may crosscut other aspectual or non-aspectual components by adding a new state or behavior or modifying the existent behavior. AOGA has been used in the context of agent orientation to describe crosscutting agent internal architecture concerns [10] (e.g. interaction, adaptation, autonomy, knowledge, collaboration, roles, learning and mobility). We admit that it is not mandatory to implement a system designed as MAS by using a MAS platform. In fact, there are some cases in which, in the same system, the level of autonomy of each agent differs or the degree of learning ability varies from one agent to another. In these cases, the approach presented in [10] can be highly useful to implement these individual features of agents using aspects. However, in other cases, either this situation does not happen or the effort to implement these agents is quite big and, in these cases, using a suitable agent implementation platform is quite desirable. This platform should support agent properties (autonomy, interaction, learning, etc.) as internal, i.e., already built-in. This is the case we assume in this paper and, for this reason, we do not need to worry about the separation (and latter composition) of crosscutting agent internal architecture concerns. We are interested in separating and composing the application crosscutting concerns.

The framework introduced in [2] relies on the Aspectual Goal-Oriented Language, a domain-specific language which adds aspectual abstractions in MAS design to separate and compose crosscutting MAS goals. Our ADL was conceived to allow derivation of architectural design from aspectual i* models [1], that also separates and composes crosscutting MAS goals.

Boucké and Holvoet [4] propose an approach for view composition in multi-agent architectures by extending a general-purpose architectural description language called xADL [7]. The approach introduces three relations between views (unification, mapping and refinement) to the view composition. This proposal considers agents as active objects with an interconnection mechanism. We define agents as social

entities which pursue goals, perform plans, play roles in an organization and cooperate to reach a common goal. The idea about view composition is currently being investigated to be incorporated in our approach.

6. Conclusion

We have introduced a UML profile that describes multiagent systems architecture by separating and modularizing crosscutting concerns using aspectual abstractions. Besides supporting architectural phase of Tropos with proper modeling language, our purpose was to provide a complete aspect-oriented methodology for MAS development, combining our architectural modeling language with other existing aspect-oriented approaches, such as [1, 21, 22].

Developing real case studies and performing empirical evaluations are needed to validate the completeness of the constructs and the scalability of the produced models. Currently, we are also investigating the use of MDD to generate aspectual detailed design models [21] from aspectual architectural design models (proposed in this paper) and these latter from aspectual i* models [1] in MAS development using Tropos. Most concepts involved in these three levels of abstraction are quite aligned, allowing to easily creating rules to perform vertical model transformations.

7. Acknowledgments

This work was supported by CNPq and CAPES research grants and BIT initiative.

8. References

[1] F. Alencar, et al., "Integration of Aspects with i* Models", *Agent-Oriented Information Systems IV*, LNCS, Vol. 4898, Springer-Verlag, 2008, pp. 183-201.

[2] M. Amor, A. Garcia, L. Fuentes, "AGOL: An Aspect-Oriented Domain-Specific Language for MAS", *Early Aspects at ICSE'07*, IEEE Computer Society,USA, 2007,p.4.

[3] T. Batista, C. Chavez, A. Garcia, U. Kulesza, et al., "Aspectual Connectors: Supporting the Seamless Integration of Aspects and ADLs", *20th Brazilian Symposium on Software Engineering*, SBC, Brazil, 2006, pp. 17-32.

[4] N. Boucké, T. Holvoet, "View composition in multi-agent architectures", *International Journal of Agent-Oriented Software Engineering*, 2008, pp. 3-33.

[5] J. Castro, M. Kolp, J. Mylopoulos, "Towards Requirements-Driven Information Systems Engineering: The Tropos Project", *Information Systems Journal*, 27(6), Elsevier, 2002, pp. 365-389.

[6] P. Clements, L. Northrop, *Software Architecture: An Executive Overview*. Tech. Rep. CMU/SEI-96-TR-003, 1996.

[7] E. Dashofy, et al., "A comprehensive approach for the development of modular software architecture description languages", *ACM Transactions on Software Engineering and Methodology*, 14(2), 2005, pp. 199-245.

[8] Dijkstra, E., *A Discipline of Programming*, Prentice-Hall, 1976.

[9] F. France, D. Kim, S. Ghosh, E. Song, "A UML-Based Pattern Specification Technique", *IEEE Transactions on Software Engineering*, 30(3), 2004, pp. 193-206

[10] Garcia, A., *From Objects to Agents: An Aspect-Oriented Approach*. PhD Thesis, Computer Science Department, PUC-Rio, Rio de Janeiro, Brazil, 2004.

[11] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.Loingtier, J.Irwin, "Aspect-Oriented Programming", *11th European Conf. on OO Programming*, Finland. LNCS, Vol. 1241. Springer-Verlag, 1997, pp. 220-242.

[12] D. Kim, R. France, S. Ghosh, E. Song, "Using Role-Based Modeling Language as Precise Characterizations of Model Families", *8th IEEE Intl. Conf. on Engineering of Complex Computer Systems*, Maryland, USA, 2002, p. 107.

[13] U. Kulesza, A. Garcia, C. Lucena, "Towards a Method for the Development of Aspect-Oriented Generative Approaches", *Ws on Early Aspects at OOPSLA'04*, Vancouver, Canada, 2004.

[14] Object Management Group (OMG): UML 2.0 Superstructure, http://www.omg.org/docs/formal/05-07-04.pdf, 2005, Last access 01/2009.

[15] Object Management Group (OMG): Model Driven Architecture, http://www.omg.org/mda/, Last access 01/2009.

[16] M. Pinto, L. Fuentes, J. Troya, "DAOP-ADL: An Architecture Description Language for Dynamic Component and Aspect-Based Development", *Generative Programming and Component Eng.*,48, Springer-Verlag, 2003,pp.118-137.

[17] M. Pinto, L. Fuentes, "AO-ADL: An ADL for describing Aspect-Oriented Architectures", *10th Intl. Ws on Early Aspects at AOSD'07*, Vancouver, Canada, 2007.

[18] A. Rashid, A. Moreira, J. Araújo, "Modularisation and Composition of Aspectual Requirements", *2nd Intl. Conf. on Aspect-Oriented Soft. Development*, USA, 2003, pp. 11-20.

[19] C. Silva, et al., "Improving Multi-Agent Architectural Design", *Software Engineering for Multi-Agent Systems V: Research Issues and Practical Applications*, LNCS, Vol. 4408, Springer-Verlag, 2007, pp. 165-184.

[20] C. Silva, et al. "Designing Social Patterns using Advanced Separation of Concerns", *CAiSE'07*, Norway, LNCS, Vol. 4495, Springer-Verlag, 2007, pp. 309-323.

[21] C. Silva, et al., "A Modeling Language for Advanced Separation of Concerns in Multi-Agent Systems", *Iberoamerican Ws on Requirements Engineering and Software Environments*, 2008, Recife, Brazil, pp. 267-280.

[22] C. Silva, et al., "Advanced Separation of Concerns in Agent Oriented Design Patterns", *International Journal of Agent-Oriented Software Engineering*, 2009 (To appear).

[23] E. Yu, "Towards modeling and reasoning support for early requirements engineering", IEEE Intl. Symposium on Requirements Engineering, USA, 1997, pp. 226-235.